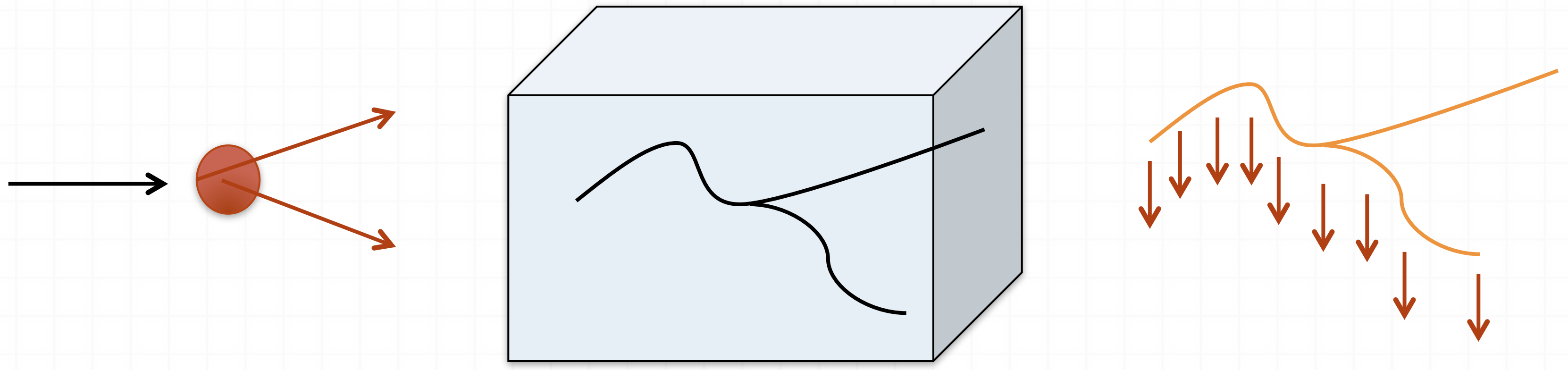
Physics Lists N Stuff

Ben Jones, MIT

Physics Lists

- A physics list is an object in LArG4 which tells Geant4 which physics processes to enable when particles are stepping through the detector volume.



1. Neutrino nucleus interaction (event generator)

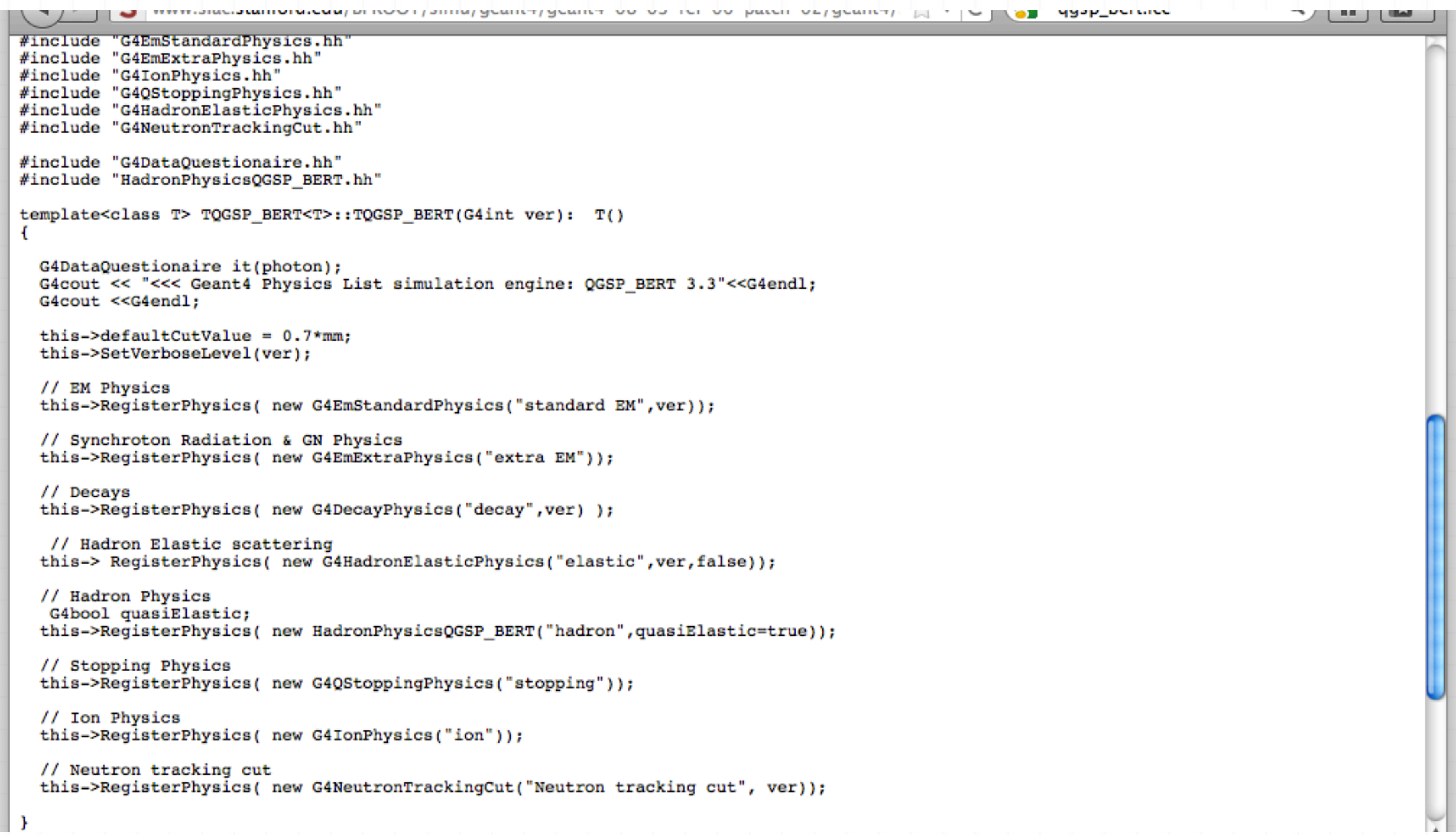
2. Stepping particles through detector (LArG4)

3. Charge drift, electronics simulations, etc

What Does a Physics List Do?

- A physics list defines all particles and interactions which Geant4 can track.
- Every process attached to a particle has a characteristic step length, this determines how far each step in the simulation is
- Hence no physics processes = no particle stepping
- Particles and interactions are organized into smaller sub-units called physics constructors.
- The job of the physics list is to register the appropriate set of physics constructors and hence select the required particles and interactions for the simulation

QGSP_BERT



```
#include "G4EmStandardPhysics.hh"
#include "G4EmExtraPhysics.hh"
#include "G4IonPhysics.hh"
#include "G4QStoppingPhysics.hh"
#include "G4HadronElasticPhysics.hh"
#include "G4NeutronTrackingCut.hh"

#include "G4DataQuestionaire.hh"
#include "HadronPhysicsQGSP_BERT.hh"

template<class T> TQGSP_BERT<T>::TQGSP_BERT(G4int ver):  T()
{

  G4DataQuestionaire it(photon);
  G4cout << "<<< Geant4 Physics List simulation engine: QGSP_BERT 3.3"<<G4endl;
  G4cout <<G4endl;

  this->defaultCutValue = 0.7*mm;
  this->SetVerboseLevel(ver);

  // EM Physics
  this->RegisterPhysics( new G4EmStandardPhysics("standard EM",ver));

  // Synchroton Radiation & GN Physics
  this->RegisterPhysics( new G4EmExtraPhysics("extra EM"));

  // Decays
  this->RegisterPhysics( new G4DecayPhysics("decay",ver) );

   // Hadron Elastic scattering
  this-> RegisterPhysics( new G4HadronElasticPhysics("elastic",ver,false));

  // Hadron Physics
   G4bool quasiElastic;
  this->RegisterPhysics( new HadronPhysicsQGSP_BERT("hadron",quasiElastic=true));

  // Stopping Physics
  this->RegisterPhysics( new G4QStoppingPhysics("stopping"));

  // Ion Physics
  this->RegisterPhysics( new G4IonPhysics("ion"));

  // Neutron tracking cut
  this->RegisterPhysics( new G4NeutronTrackingCut("Neutron tracking cut", ver));

}
```

Physics Constructors

- The first thing a physics constructor does is declare all the particles it applies to
- Then for each type of particle, a singleton process manager is passed instructions on which physics processes apply to that particle
- For example, some lines from G4EmPhysicsStandard:

```
// gamma
G4Gamma::Gamma();

// leptons
G4Electron::Electron();
G4Positron::Positron();
G4MuonPlus::MuonPlus();
G4MuonMinus::MuonMinus();
.....
```

```
if (particleName == "gamma") {

pmanager->AddDiscreteProcess(new G4PhotoElectricEffect);
pmanager->AddDiscreteProcess(new G4ComptonScattering);
pmanager->AddDiscreteProcess(new G4GammaConversion);

} else if (particleName == "e-") {

......
```

Geant4 InBuilt Physics Lists

- Geant4 features several sample physics lists.
- In general all feature the same electromagnetic physics – which is the majority of what we care about, including:
  - Ionization, coulomb scattering, bremstrahlung, pair production, e+ absorption, multiple scattering, etc etc
- Hadronic physics varies a lot between physics lists. For our events I think it is true to say that this only ever becomes important at very low energy?
- * Mention G4LowEnergyEM

Examples of Physics Lists

- LHEP, QGSP, QGSC, FTFP, FTFC : “LHEP” : parameterized “QGS” : Quark Gluon String; “FTF”: Fritjof; “P”: Pre-equilibrium “C”: CHIPS.
- n xxxx_BERT, xxxx_BIC : intra-nuclear transport models, Bertin and binary cascade.
- nn xxxx_GN : photon-nuclear reactions n xxxx_HP : high precision low-energy
- neutron transportation n xxxx_LEAD : leading-particle biasing.

- LArSoft default is QGSP_BERT – this chosen by Bill S, I believe to mirror the ATLAS LAr calorimeter physics list

Configurability of LArSoft physics list

- For most simulation applications, a basic physics list is chosen and then tweaked to meet the needs of the experiment at hand.
- In LArSoft we need some more flexibility – for some simulation jobs we want to simulate optical photons, but for others this is unnecessary
- Running optical simulations carries a high computational pricetag – a few hours per event.
- Hence we also have a second, library sampling, much faster optical simulation
- Which optical physics to run (if any) is down to the physics list – hence we want control of this on a job-by-job basis

Setting the Physics List

- The LArSoft physics list is built to allow users to turn on or off any Geant4 physics constructor at run time.
- It works like this:
- Exerpt from simulationservices.fcl:

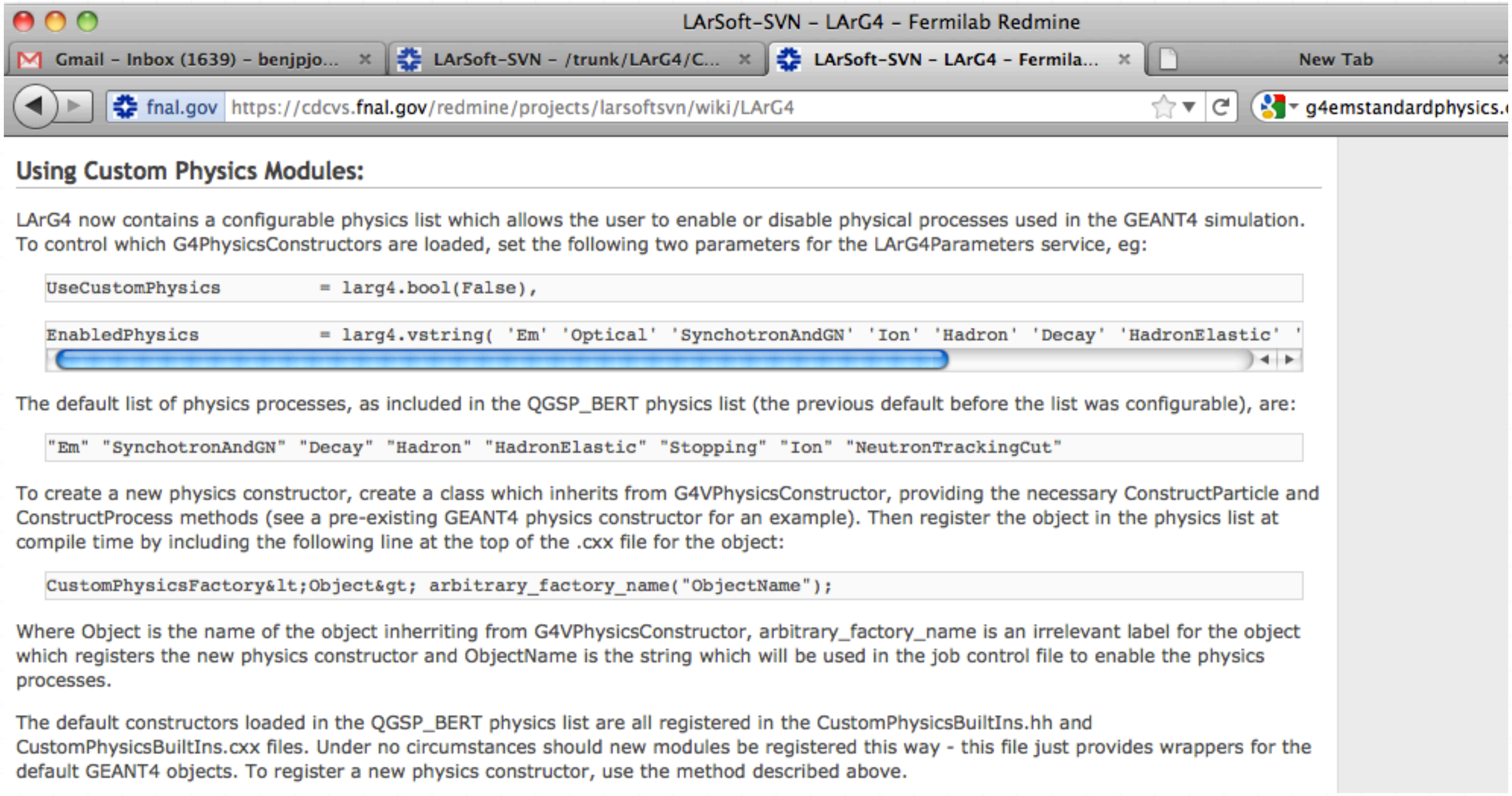
```
UseCustomPhysics:       false   #Whether to use a custom list of physics processes or the default
EnabledPhysics:        [ "Em", "SynchrotronAndGN", "Ion", "Hadron",
                         "Decay", "HadronElastic", "Stopping", "NeutronTrackingCut" ]
```

Names of physics constructors ^^

Physics Constructors

- Some of the physics constructors allowed are default Geant4 classes, whereas some are custom assembled for larsoft (eg optical physics classes).
- The physics list implementation is based on the “factory” design scheme, whereby the set of accessible physics constructors is not hard coded in the physics list implementation
- New physics constructors are registered at compile-time and no code should be added to the physics list class ever.
- Since this was implemented, various hacks have been added which violate this design scheme– I am planning to tidy these up soon.

Adding a custom physics constructor



LArSoft-SVN – LArG4 – Fermilab Redmine

Gmail – Inbox (1639) – benjpjo... | LArSoft-SVN – /trunk/LArG4/C... | LArSoft-SVN – LArG4 – Fermila... | New Tab

fnal.gov https://cdcvs.fnal.gov/redmine/projects/larsoftsvn/wiki/LArG4

g4emstandardphysics.

Using Custom Physics Modules:

LArG4 now contains a configurable physics list which allows the user to enable or disable physical processes used in the GEANT4 simulation. To control which G4PhysicsConstructors are loaded, set the following two parameters for the LArG4Parameters service, eg:

```
UseCustomPhysics          = larg4.bool(False),
```

```
EnabledPhysics            = larg4.vstring( 'Em' 'Optical' 'SynchotronAndGN' 'Ion' 'Hadron' 'Decay' 'HadronElastic' '
```

The default list of physics processes, as included in the QGSP_BERT physics list (the previous default before the list was configurable), are:

```
"Em" "SynchotronAndGN" "Decay" "Hadron" "HadronElastic" "Stopping" "Ion" "NeutronTrackingCut"
```

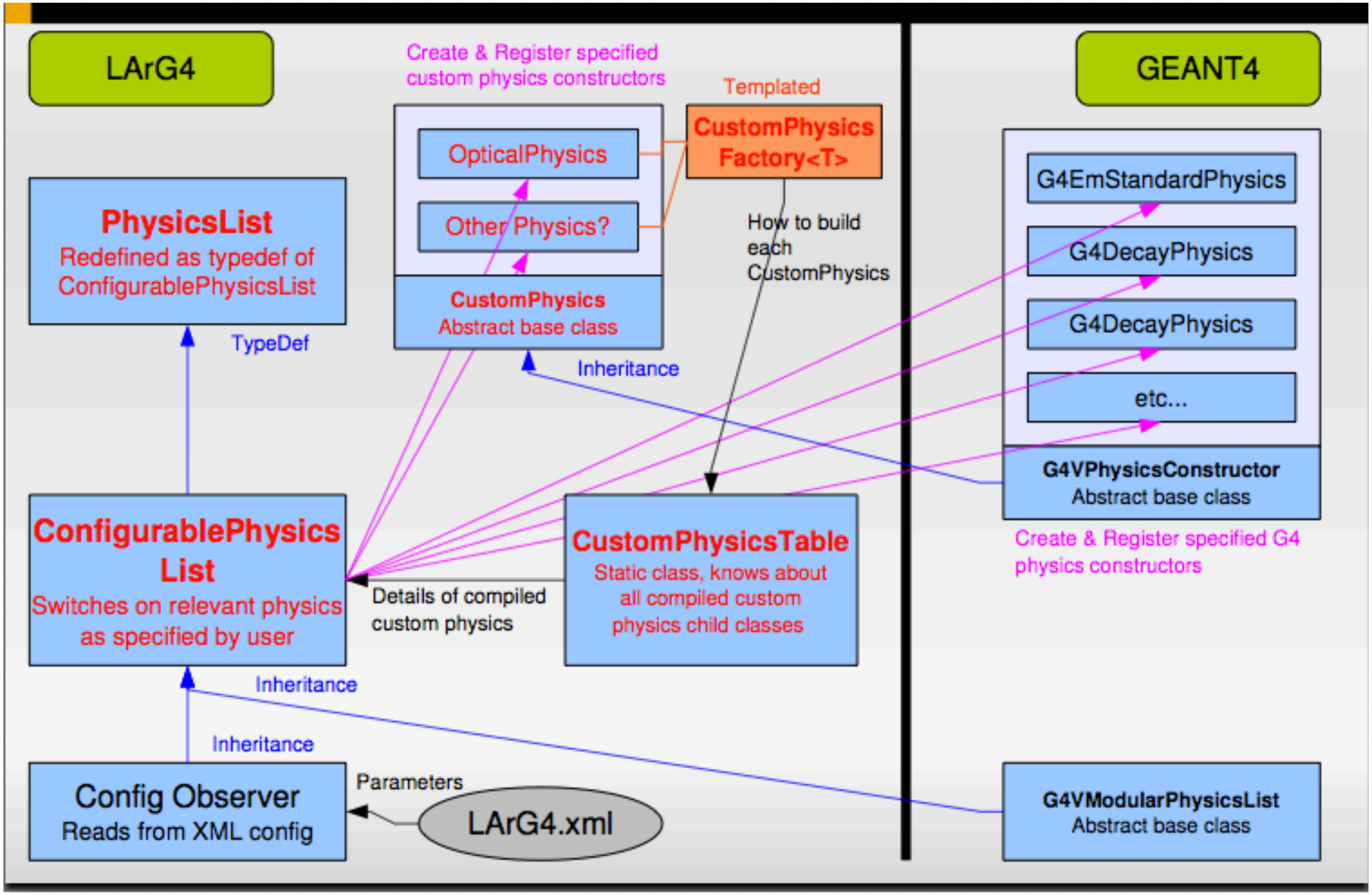
To create a new physics constructor, create a class which inherits from G4VPhysicsConstructor, providing the necessary ConstructParticle and ConstructProcess methods (see a pre-existing GEANT4 physics constructor for an example). Then register the object in the physics list at compile time by including the following line at the top of the .cxx file for the object:

```
CustomPhysicsFactory&lt;Object&gt; arbitrary_factory_name("ObjectName");
```

Where Object is the name of the object inherriting from G4VPhysicsConstructor, arbitrary_factory_name is an irrelevant label for the object which registers the new physics constructor and ObjectName is the string which will be used in the job control file to enable the physics processes.

The default constructors loaded in the QGSP_BERT physics list are all registered in the CustomPhysicsBuiltIns.hh and CustomPhysicsBuiltIns.cxx files. Under no circumstances should new modules be registered this way - this file just provides wrappers for the default GEANT4 objects. To register a new physics constructor, use the method described above.

The Guts


LArG4
Create & Register specified custom physics constructors
Templated
CustomPhysics Factory<T>
GEANT4
OpticalPhysics
Other Physics?
CustomPhysics
Abstract base class
PhysicsList
Redefined as typedef of ConfigurablePhysicsList
TypeDef
How to build each CustomPhysics
G4EmStandardPhysics
G4DecayPhysics
G4DecayPhysics
etc...
Inheritance
G4VPhysicsConstructor
Abstract base class
Create & Register specified G4 physics constructors
ConfigurablePhysics List
Switches on relevant physics as specified by user
Details of compiled custom physics
CustomPhysicsTable
Static class, knows about all compiled custom physics child classes
Inheritance
Inheritance
Config Observer
Reads from XML config
Parameters
LArG4.xml
G4VModularPhysicsList
Abstract base class

Registering a module

- No code added to the physics list class to add new module, rather:
- Custom physics class:

  Eg in OpticalPhysics.cxx

  ```
  CustomPhysicsFactory<OpticalPhysics> optical_factory("Optical");
  ```
- Existing Geant4 physics class are wrapped in CustomPhysicsFactories in the files

  CustomPhysicsBuiltIns.hh ***CustomPhysicsBuiltIns.cxx***